AD-A008 358

SOLVING SINGULARLY CONSTRAINED TRANSSHIPMENT PROBLEMS

Fred Glover, et al

Texas University at Austin

Prepared for:

Naval Personnel Research and
Development Laboratory
Office of Naval Research

December 1974

Unclassified
Security Classification

AD-A008 358

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Center for Cybernetic Studies The University of Texas	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE
Solving Singularly Constrained Transshipment Problems

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)
Fred Glover
D. Karney
D. Klingman
R. Russell

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1974	29	34

8a. CONTRACT OR GRANT NO. N00123-74-C-2275	9a. ORIGINATOR'S REPORT NUMBER(S) Center for Cybernetic Studies Research Report CCS 212
b. PROJECT NO. c. d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Naval Research (Code 434) Washington, D.C.

13. ABSTRACT

This paper develops a primal simplex procedure to solve transshipment problems with an arbitrary additional constraint. The procedure incorporates efficient methods for pricing-out the basis, determining representations, and implementing the change of basis. These methods exploit the near triangularity of the basis in order to take full advantage of the computational schemes and list structures used in solving the pure transshipment problem. We also report the development of a computer code, I/O PNETS-I for solving large scale singularly constrained transshipment problems. This code has demonstrated its efficiency over a wide range of problems and has succeeded in solving a singularly constrained transshipment problem with 3000 nodes and 12,000 variables in less than 5 minutes on a CDC 6600. Additionally, a fast method for determining near optimal integer solutions is also developed. Computational results show that the near optimum integer solution value is usually within a half of one percent of the value of the optimum continuous solution value.

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Transportation problems						
Transshipment problems						
Constrained problems						
Linear Programming						
Computational L. P. testing						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0102-014-6800

ii

Unclassified
Security Classification

Research Report CCS 212

SOLVING SINGULARLY CONSTRAINED TRANSSHIPMENT PROBLEMS

by

Fred Glover*
D. Karney**
D. Klingman***
R. Russell****

December 1974

* Professor, University of Colorado
** Research Associate, University of Texas
*** Professor, University of Texas
**** Professor, University of Tulsa

This research was partly supported by the Navy Personnel Research and Development Laboratory Contract N00123-74-C-2275 with the Center for Cybernetic Studies, The University of Texas, Austin, Texas. Reproduction in part or in whole is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 512
The University of Texas
Austin, Texas 78712
(512) 471-1821
(512) 471-4894

Abstract

This paper develops a primal simplex procedure to solve transshipment problems with an arbitrary additional constraint. The procedure incorporates efficient methods for pricing-out the basis, determining representations, and implementing the change of basis. These methods exploit the near triangularity of the basis in order to take full advantage of the computational schemes and list structures used in solving the pure transsipment problem. We also report the development of a computer code, I/O PNETS-I for solving large scale singularly constrained transshipment problems. This code has demonstrated its efficiency over a wide range of problems and has succeeded in solving a singularly constrained transshipment problem with 3000 nodes and 12,000 variables in less than 5 minutes on a CDC 6600. Additionally, a fast method for determining near optimal integer solutions is also developed. Computational results show that the near optimum integer solution value is usually within a half of one percent of the value of the optimum continuous solution value.

1. INTRODUCTION

In this paper we consider the capacitated transshipment problem which contains an additional linear constraint. The pure transshipment problem is well known for its wide range of practical application and for the facility with which it can be solved. Many linear programming models are formulated as transshipment problems in order to gain computational efficiency. Researchers have developed special purpose transshipment algorithms that are roughly 150-300 times faster than general purpose linear programming codes [2,17,24,30,32] for solving transshipment problems.

Unfortunately, this significant gain in efficiency is usually lost upon the addition of a single extra constraint. In related constrained transportation problems, researchers have investigated certain classes of extra constraints which allow a transformation of the singularly constrained transportation problem to an enlarged equivalent transportation problem [3,4,5,6,7,26, 33]. The paper [26] describes similar transformation techniques in a transshipment context. Many constrained network problems, however, cannot be transformed into a transportation or transshipment problem, and thus a technique for solving any constrained transshipment problem efficiently is needed.

In this paper, a specialized primal simplex procedure is developed which is applicable to capacitated transshipment problems with an arbitrary extra constraint. The procedure is a specialization of the algorithms developed in [4,27,28,29] for handling an arbitrary number of extra constraints. Computational results indicate that our solution procedure is at least 75 times faster than state-of-the-art general purpose linear programming codes in solving this class of problems.

The addition of an arbitrary additional constraint may destroy the integrality of the solution to a transshipment problem. Thus, a very simple and fast procedure is proposed for determining a near optimal integer solution for such problems, whenever the additional constraint is an inequality.

There are a number of applications which lie within the domain of singularly constrained capacitated transshipment problems. For example, in real-world transshipment problems, the modeller is often faced with two objectives or goals (such as, maximizing profit while minimizing shipping time). Other applications include problems that can be formulated as a transshipment problem with extra linear constraints where an integer solution is required. The code development discussed in this paper makes these problems amenable to solution via surrogate constraint techniques (see, e.g., [1,11,12,13,14]).

2. PROBLEM DEFINITION AND BASIS PROPERTIES

A capacitated singularly constrained transshipment problem may be stated as:

minimize $\sum_{(i,j)\varepsilon A} c_{ij} x_{ij} + OS$

subject to: $$-\sum_{(i,j)\varepsilon A} x_{ij} + \sum_{(j,i)\varepsilon A} x_{ji} = a_i \ , \quad i\varepsilon N \tag{1}$$

$$0 \leq x_{ij} \leq U_{ij} \ , \quad (i,j)\varepsilon A \tag{2}$$

$$\sum_{(i,j)\varepsilon A} f_{ij} x_{ij} \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} k \tag{3}$$

where a_i represents the supply (demand) at node i, $\sum_{i\varepsilon N} a_i = 0$, x_{ij} is the flow from node i to node j on arc (i,j), N is the set of nodes in the network, A is the set of arcs in the network, c_{ij} is the cost on arc (i,j), and U_{ij} is the upper bound on arc (i,j). (Note that the f_{ij} in (3) may be positive, negative, or zero.)

The dual problem is:

$$\text{maximize} \quad \sum_{i \in N} a_i w_i - \sum_{(i,j) \in A} w_{ij} U_{ij} + k\delta$$

$$\text{subject to:} \quad -w_i + w_j - w_{ij} + f_{ij}\delta \leq c_{ij} \quad , (i,j) \in A \tag{4}$$

$$w_i \text{ unrestricted, } i \in N$$

$$w_{ij} \geq 0 \qquad (i,j) \in A$$

$$(\delta \geq 0, \quad \delta\text{-unrestricted,} \quad \delta \leq 0 \text{ depending on } (3))$$

Graphically, the problem may be viewed as a network flow graph whose arcs have "flags" (i.e., additional numerical values) corresponding to the coefficients in constraint (3). It is generally known that a basis for a capacitated transshipment problem consists of n-1 arcs which span the network containing n nodes [7,23]. The addition of constraint (3) to the problem normally increases the rank of the "incidence" matrix by one, thus requiring the use of n arcs in the basis of the singularly constrained transshipment problem. (In the case where the rank is not increased, constraint (3) may be deleted without loss of generality if the system is consistent.) Clearly n-1 of these n arcs will form a spanning tree with the network flow graph since all n nodes must be spanned. A fundamental property of spanning trees is that the addition of one additional arc incident on the tree nodes forms precisely one closed loop with attached trees [7,23,27,28]. Thus, the basis for the singularly constrained transshipment problem may be characterized as a spanning tree plus one additional arc. (Note that the additional arc may be the slack or artificial variable S from constraint (3).)

We will show how to exploit this near triangular basis structure, to yield a computationally efficient primal simplex procedure for solving such problems. This specialized simplex approach yields an inverse compactification which greatly reduces the basis information that has to be stored between successive iterations and that correspondingly reduces the arithmetic calculations required in pivoting.

3. PRICING-OUT THE BASIS

In this section we present special procedures for determining the dual evaluator values and the updated costs (or $z_j - c_j$ values). Pricing out the basis is equivalent to finding the simultaneous solution of the equality form of the dual constraints in (4) associated with the primal variables in the basis. Our procedure is a direct extension of the pricing-out procedure given in [16,20] for the unconstrained or pure transshipment problem. The procedure is a two step approach that first obtains the value of the dual evaluator δ and subsequently determines the remaining dual evaluators w_i.

A value is obtained for the dual evaluator δ by making use of the fact that the basis may be partitioned and stored as a spanning tree plus an additional arc, as depicted in Figure 1. Thus, the arcs may be stored via the efficient list structure of [16,20] for maintaining and updating spanning trees. The loop present in a basis for the constrained transshipment problem will be referred to subsequently as the basis loop.

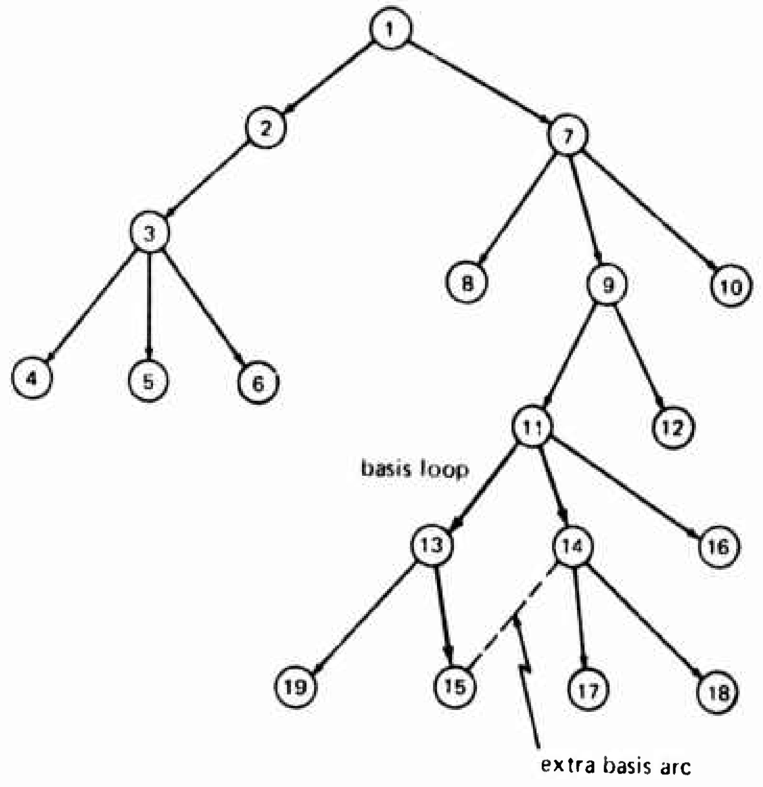

Figure 1. A possible basic for the singularly constrained transshipment problem.

If we let B represent the basis arcs, then applying complementary slackness to the dual system (4) we have:

$$-w_i + w_j + \delta f_{ij} = c_{ij} \quad , \ (i,j)\varepsilon\ B \tag{5}$$

Thus, once δ is known, the remaining dual evaluators can be determined immediately by setting the root node's dual evaluator w_i to 0, and proceeding downward through the spanning tree using (5) to determine the remaining dual evaluators. (Note that the w_{ij} in (4) are omitted in (5) since they may be arbitrarily set equal to 0 for the basis arcs). We now give a simple formula for computing the value of δ relative to a given basis.

Remark 1:

An explicit solution value for δ associated with any particular basis is:

$$\delta = \frac{\sum_F c_{ij} - \sum_R c_{ij}}{\sum_F f_{ij} - \sum_R f_{ij}} \tag{6}$$

where F is the set of arcs traversed in the forward direction in going from any node on the basis loop back to itself, and R is the set of arcs traversed in the reverse direction.

Proof: The proof is straightforward and only involves an examination of the subsystem of dual equations associated with arcs in the basis loop. Let node i be some node on the basis loop. The result may be established by solving for w_j in terms of δ and w_i using (5) and then solving repeatedly for each successive variable w_j on the basis loop in terms of the preceding using (5). When node i is re-encountered and the resulting w_i is substituted into (5), equation (6) is derived.

Note the ease with which δ can be computed if the basis is stored as a spanning tree via [16,20] and an extra basis arc. In particular, only a simple trace of the basis loop is required keeping two accumulators. Having determined the value of δ, the remaining dual evaluators can be easily determined (as indicated earlier) by proceeding downward through the spanning tree.

Using Remark 1, it is clear that the value of δ changes from basis to basis if and only if the basis loop changes. Thus, the previously described pricing-out procedure is only required at those iterations in which the arc leaving the basis is also in the basis loop. If, however, the leaving arc is not in the basis loop, then δ does not change and it is not necessary to update δ or all of the w_i dual evaluators.

To be precise, suppose arc (r,s) is to enter the basis. The updated cost of this entering arc is $\bar{c}_{rs} = c_{rs} + w_r - w_s - f_{rs}\delta$. The new updated w_r's values can be obtained by adding $\pm \bar{c}_{rs}$ to one of the two subtrees created when the arc leaving the basis is deleted. Normally the subtree not containing the root node of the basis tree [16] would be updated. More specifically, if node r is a member of the disjoint subtree to be updated, then $\bar{c}_{rs}$ is subtracted from all w_i's associated with nodes in this disjoint subtree. However, if node s is a member of this subtree then $\bar{c}_{rs}$ is added to all w_i's associated with nodes in the subtree. The validity of this procedure follows directly from the fact that (5) holds for the new basis using the updated w_i values. That is, (5) holds for arcs in the basis contained in the non-updated subtree since (5) held for these arcs in the old basis. Further, (5) holds for the entering arc (r,s) and the arcs in the updated subtree by construction.

4. CHANGE OF BASIS PROCEDURES

This section characterizes procedures for determining the vector representations of the arc entering the basis, the vector to leave the basis, and the updated basic flows. We provide a connection between the calculation of δ in the preceding section and calculations involved in carrying out the remaining change of basis steps, thus, enabling the latter steps to be carried out with marginal additional effort.

First, consider the determination of the representation of the arc (r,s) which is to enter the basis. Let $\hat{P}_{rs}$ denote the column of the incidence matrix associated with arc (r,s) and let P_{rs} denote that portion of $\hat{P}_{rs}$ associated with the underlying pure transshipment problem; $\hat{P}_{rs}$ is an n+1 component vector, whereas P_{rs} is an n component vector.

Two observations lead to an efficient way to determine the representation of the entering arc (r,s). The first n components of the representation of

$\hat{P}_{rs}$ may be obtained from a linear combination (consisting only of +1's and -1's) of certain arcs in the basis tree. These arcs correspond to the arcs in the loop created when arc (r,s) is added to the current basis tree. See Figure 2.

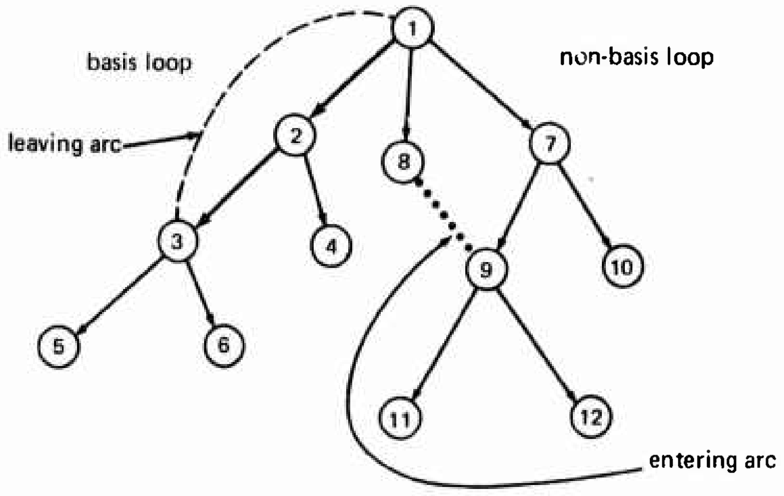

Figure 2. Illustration of basis and non-basis loops.

The last component of the representation of $\hat{P}_{rs}$ may be obtained from a linear combination (consisting of +1's and -1's) of the arcs in the current basis loop. Let the arcs (i_1,i_2), $(i_2,i_3),\ldots,(i_p,i_1)$ comprise the current basis loop.

Then clearly

$$+1\ \hat{P}_{i_1} - 1\hat{P}_{i_2} + 1\hat{P}_{i_3} + \ldots - \hat{P}_{i_p} = \begin{pmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ \sum\limits_{(i,j)\varepsilon F} f_{ij} - \sum\limits_{(i,j)\varepsilon R} f_{ij} \end{pmatrix} \qquad (7)$$

where F and R are the sets of forward and reverse arcs defined earlier. If we let NF and NR, respectively, denote the set of arcs traversed in the forward direction and the set of arcs traversed in the reverse direction in the non-basis loop in going from node r to node s, then we can determine the representation of $\hat{P}_{rs}$ via

$$\begin{pmatrix} P_{rs} \\ \sum_{(i,j)\varepsilon NF} f_{ij} - \sum_{(i,j)\varepsilon NR} f_{ij} \end{pmatrix} + \theta \begin{pmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ \sum_{(i,j)\varepsilon F} f_{ij} - \sum_{(i,j)\varepsilon R} f_{ij} \end{pmatrix} = \begin{pmatrix} P_{rs} \\ f_{rs} \end{pmatrix} = \hat{P}_{rs} \qquad (8)$$

which implies

$$\theta = \frac{f_{rs} - \sum_{(i,j)\varepsilon NF} f_{ij} + \sum_{(i,j)\varepsilon NR} f_{ij}}{\sum_{(i,j)\varepsilon F} f_{ij} - \sum_{(i,j)\varepsilon R} f_{ij}} \qquad (9)$$

Note that the denominator in (9) is known from the δ calculation. The denominator in (9) is also non-zero, since the arcs in the current basis loop are linearly independent. Thus, the representation of the entering arc is obtained by attaching +1's and -1's to the arcs in the non-basis loop and θ's and $-\theta$'s to the arcs in the basis loop. In the special cases where the slack or artificial variable S is entering or leaving the basis, expressions (8) and (9) are further simplified by means of analogous ideas. (We omit these simplifications for the sake of brevity. The working paper [28] describes these simplifications in detail for constrained transportation problems.)

In determining the arc to leave the basis and the updated flows, there are essentially two cases. In case 1 the basis loop and the non-basis loop have no arcs in common; in case 2 the loops overlap.

Case 1(a): $\theta = 0$

The arc to leave the basis is in the non-basis loop and the pivot step is identical to a pivot for a pure transshipment problem. There are two subcases depending on whether the arc entering the basis is at its upper or lower bound.

Case 1(b): $\theta \neq 0$

Minimum flow change values must be computed for both loops and the arc associated with the smaller of these minimum flow change values is selected to leave the basis. There are four possible subcases depending on whether θ is less than zero or greater than zero, and whether the entering arc is at its upper or lower bound.

Case 2(a): $\theta = 0$

Same as case 1(a).

Case 2(b): $\theta \neq 0$

The minimum flow change must be computed on each loop's non-overlapping arcs and on the overlapping arcs. These minimum flow change values depend on the signs of θ, $\theta + 1$, and the condition of the arc entering the basis (i.e., whether it is at its upper or lower bound).

It is important to note that the calculation of θ and the determination of the arc to leave the basis is made particularly easy by keeping the basis stored as a spanning tree and an extra basis arc. Additionally, once the flow change value γ has been determined, the flows can be quickly updated by traversing the non-basis loop one more time and adding or subtracting γ from its current flow depending on the direction and "bound condition" of the arc. Similarly, the flows on the basis loop can be updated by traversing arcs and adding or subtracting $\theta\gamma$ as appropriate.

5. BASIS PARTITIONING

The partitioning of the basis into a spanning tree and an extra arc is quite easy to maintain from basis to basis. The partitioning is automatically preserved if the arc leaving the basis is not contained in the basis loop; that is, in this case, the entering arc may be pivoted into the spanning tree segment of the basis in the same manner as for a pure transshipment problem [16,20].

When the arc leaving the basis is contained in the basis loop and is not the extra basis arc, the basis partitioning can be preserved simply by pivoting the current extra basis arc into the spanning tree in place of the arc leaving the basis. The arc entering the basis then becomes the new extra basis arc.

Finally, if the arc leaving the basis is the extra basis arc, then the entering arc simply becomes the new extra arc and no further updating is required. The validity of this procedure is established by observing that the entering arc and the current spanning tree will never yield a loop in the new hypothesized spanning tree since an arc contained in the loop created by the entering arc in the old spanning tree is always being deleted.

6. BASIC STARTING SOLUTIONS

A basic "feasible" solution for the singularly constrained transshipment problem may be obtained by applying any basic starting method for a transshipment problem [7,15,24] and then adding an appropriate slack or artificial variable (as determined by equation (3)) to this spanning tree. Another way of obtaining a good basic start is to use a basic optimal solution to the underlying transshipment problem.

In this paper, we have tested two or three versions of each of four distinct starting rule procedures. One of these uses the optimal solution to the underlying transshipment problem. Another uses the modified row minimum start [15] which has proven to be computationally best for solving pure transshipment problems. The third procedure uses the modified row minimum start but gives priority to arcs with a positive f_{ij} if (3) is a "greater than or equal to" constraint. The fourth method uses a Lagrangean relaxation approach [8,9,11,22,31] to obtain a basic optimal solution to a pseudo-transshipment problem. These starts will be described in more detail in the next section.

7. INTEGER SOLUTION

The optimal solution to a singularly constrained transshipment problem may not be integer valued even if the parameters are integers. However, if the additional constraint (3) is an inequality constraint, an integer solution (except possibly for S) may be obtained by simply pivoting the slack variable S into the optimal basis. If the right hand side and capacity parameters of the underlying transshipment problem are integers, the resulting solution will be integer valued in the x_{ij} since the basic x_{ij} corresponds to an extreme pivot of the underlying transshipment problem.

The slack variable S may always be pivoted into the optimal basis if the problem is capacitated since the representation of the slack is non-zero. (If the problem is uncapacitated it can be capacitated without loss of generality in most practical settings by restricting the flow on each arc to be less than or equal to the total supply.) The results in the next section indicate that this single pivot procedure provides a good integer solution. In particular, the integer solution objective function values for the problems solved using this approach were always within .007 of the optimal continuous solution value.

8.0 COMPUTATIONAL RESULTS

8.1 INTRODUCTION

In order to test the proposed efficiency of the preceding algorithm, we designed and implemented the computer code I/O PNETS-I. The main computational routine of I/O PNETS-I is a modification of the in-core out-of-core transshipment code I/O PNET-I. I/O PNET-I is a state-of-the-art code for solving large scale transshipment problems [24]. (This code recently solved a problem with 5000 nodes and 625,000 arcs in less than 10 minutes on a CDC 6600. A modified version of I/O PNET-I, developed by Analysis, Research, and Computation, Inc., is capable of solving problems with 50,000 nodes and 62 million arcs on a CDC 6600, UNIVAC 1108, IBM 360/65, or IBM 370/155.)

The computational results reported in this section indicate that I/O PNETS-I is able to solve singularly constrained transshipment problems in approximately twice the time that I/O PNET-I can solve the underlying transshipment problem. (I/O PNET-I [24] has been shown to be 150-300 times faster than the state-of-the-art linear programming code OPHELIE/LP.) Also, I/O PNETS-I can solve singularly constrained transshipment problems which are as large as the transshipment problem handled by I/O PNET-I.

8.2 OVERVIEW OF I/O PNETS-I

The I/O PNETS-I computer code is written in FORTRAN IV. It was initially tested on a CDC 6600 with a maximum core memory of 130,000 words using the RUN compiler. The code uses the augmented threaded index method [20] to store and update the basis data. The code keeps all of the basis data in central memory which requires seven node length arrays. The arc data are stored on a sequential access disk file. The arc data are brought (paged) into central memory in accordance with a specified buffer (page) size B_A. The code keeps a list of arcs of size B_C in central memory whose flows are equal to their upper bounds but are not recorded in the arc data on the disk file. This list of arcs includes those arcs whose flows are not equal to their upper bounds but are so recorded in the arc data on the disk file. When all arrays are dimensioned to 1, the code requires 9000 words of central memory. In general, to solve a singularly constrained transshipment problem with N nodes and A arcs (without exploiting the word size of the machine) requires

$7|N| + 3B_A + 9000$ words if the problem is uncapacitated

and

$7|N| + 4B_A + B_C + 9000$ words if the problem is capacitated

where B_A is the number of arcs in the arc page buffer and B_C is the number of arcs in the capacity buffer.

It would be possible, by exploiting the fact that the costs and node numbers are integer valued, to store more than one of these data entries per word and in this manner reduce the storage requirements. However, our purpose was to develop a code whose capabilities do not depend on the unique characteristics of a particular computer (such as word size). The obvious advantage of this approach is the ease with which it enables the code to be tested on different machines. Further, a "manilla" FORTRAN IV was used so that re-coding to fit differing machine configurations would be minimized. Within these constraints, we sought to minimize storage requirements, at the same time making sure the code could solve the "thoroughly general" singularly constrained transshipment problem. Thus, for example, the code is designed to allow multiple arcs between the same nodes and to handle arbitrarily capacitated problems; thereby making it possible to accommodate piecewise linear convex cost minimization. A standard translation of variables with non-zero lower capacities is performed upon inputing the problem in order to make all such capacities equal to 0 and hence eliminate a capacity buffer for lower bounds.

The program consists of a main program and ten subroutines, and may be conceptually depicted as in boxes 1-6 in Figure 3. During the development and testing of the code 30 statistics were kept on each problem. These statistics ranged from time spent reading and writing disk records to the number of artificial arcs in the starting basis. Unfortunately, it is not possible to present all of these statistics in a concise and understandable format. Thus, we have chosen to report the following statistics: the total solution time (time spent in boxes 2-6), the number of pivots, the total pivot time (total time spent in boxes 3-6), total time and number of pivots spent in the Lagrangean search, and the total solution time and total number of pivots required to solve the underlying transshipment problem.

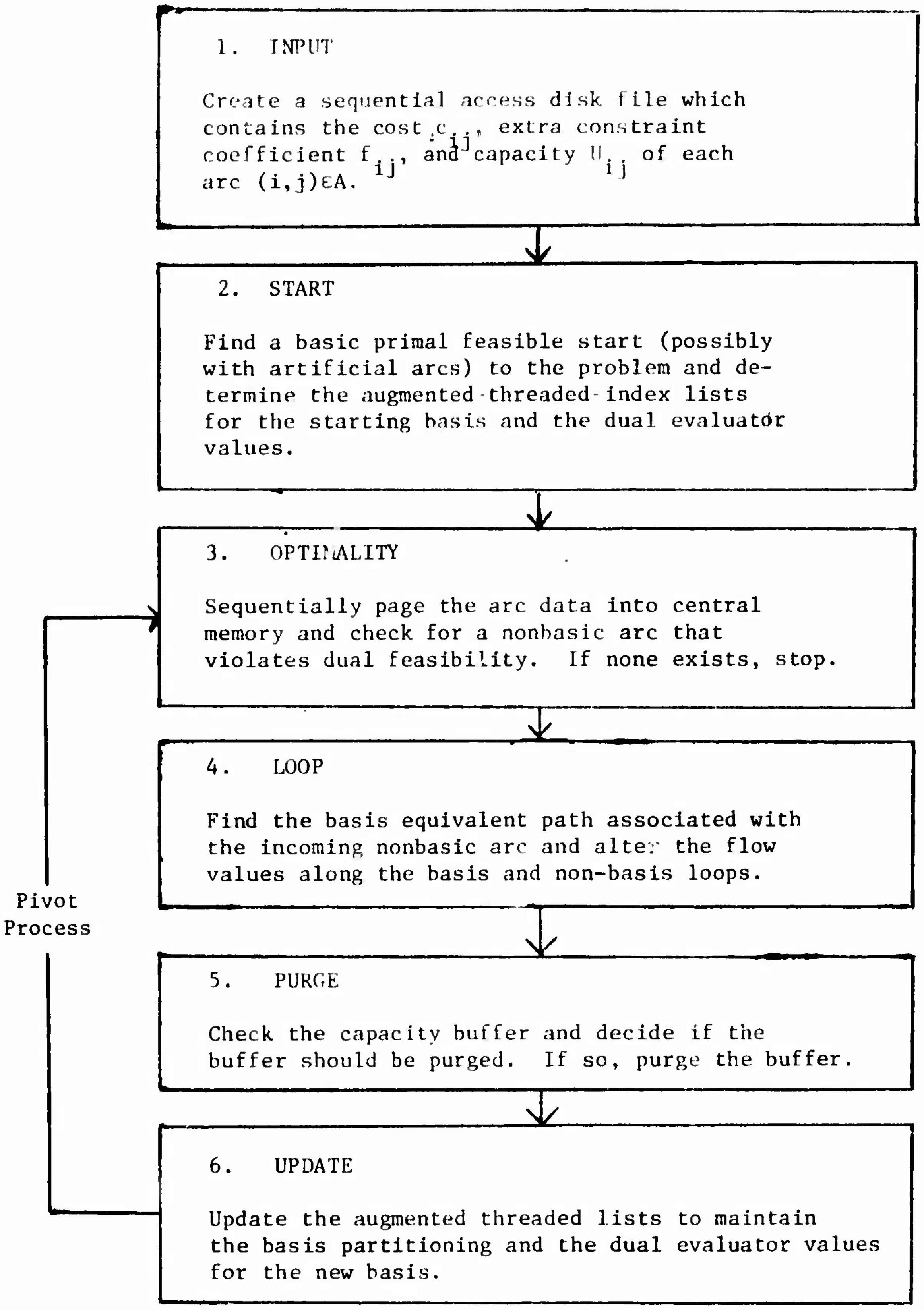
1. INPUT
Create a sequential access disk file which contains the cost c_{ij}, extra constraint coefficient f_{ij}, and capacity U_{ij} of each arc $(i,j)\varepsilon A$.
2. START
Find a basic primal feasible start (possibly with artificial arcs) to the problem and determine the augmented threaded index lists for the starting basis and the dual evaluator values.
3. OPTIMALITY
Sequentially page the arc data into central memory and check for a nonbasic arc that violates dual feasibility. If none exists, stop.
4. LOOP
Find the basis equivalent path associated with the incoming nonbasic arc and alter the flow values along the basis and non-basis loops.
5. PURGE
Check the capacity buffer and decide if the buffer should be purged. If so, purge the buffer.
6. UPDATE
Update the augmented threaded lists to maintain the basis partitioning and the dual evaluator values for the new basis.
Pivot Process

Figure 3

Flow Diagram for the In-Core Out-of-Core Primal Singularly Constrained Transshipment Code

8.3 SCOPE AND PURPOSE OF THE COMPUTATIONAL STUDY

The primary purpose of the computational study is to determine the best start and pivot rules to use in conjunction with the preceding algorithm for solving different types of singularly constrained transshipment problems. Another purpose is to evaluate the adequacy of the integer solution provided by pivoting the slack variable into the basis.

To conduct the testing seventy five feasible problems were generated. All of the problems have costs which range between 1 and 100, a total supply of 100,000, and upper capacitates ranging from 10 to 1,000. The first twenty problems have 300 nodes and 1500 arcs. The second twenty problems have 500 nodes and 2500 arcs and the third twenty problems have 1000 nodes and 5000 arcs. (The remaining fifteen problems are of varying size.) Each group of twenty problems contains the same underlying transshipment problem with four distinct extra inequality constraints (3). For each extra constraint, five different right hand side values K are given, thus producing five problems with very similar structure.

The four distinct extra constraints all contain 150 non-zero coefficients f_{ij}. One of these extra constraints has all non-zero coefficients equal to unity. Another extra constraint has non-zero coefficients that range among the integer values between 1 and 5. The third extra constraint has the same coefficient range, as the second, except the coefficients may not be integer valued. The fourth extra constraint has non-zero coefficients with values of -1 or +1.

The fifteen other problems vary in size from 1000 nodes to 3000 nodes and from 3000 arcs to 15000 arcs. The number of non-zero coefficients in the extra constraint vary from .1 to 1.0 times the number of arcs, and the values of the coefficients are similar to the preceding ones. In constrast to the first sixty problems whose extra constraints are all inequalities, some of these problems contain equality constraints.

Various combinations of starting, pivoting, and Lagrangean relaxation strategies were tested. A limitation of our stydy is that the effects of different buffers sizes and different strategies for purging the capacity buffer were not tested. Thus, the computational results in this section pertain only to problems where all problem information is kept in central memory. (These limitations are simply due to a lack of human and computer time to conduct all possible testing.) Thus, as specific applications arise, the best rules in this study should be carefully analyzed to determine their appropriateness. Recently we had an opportunity to do this on an application involving multiple objectives. In this case the extra constraint consisted of keeping a weighted aggregate of objectives above some satisfactory level. For this particular application, the rules described in this study appeared to be best.

8.4 TESTING SUMMARY

The purpose of this section is to give the reader a summary of the range of solution tactics investigated before picking a particular one to refine and streamline. In order not to overwhelm the reader with large numbers of statistics and long descriptions of fifteen different solution strategies that proved to be unsuccessful for solving the test problems efficiently, this summary will concentrate only on the computational highlights.

Table I illustrates our findings for a subset of the twenty 500 node, 2500 arc problems described in section 8.3. The extra constraint for each of these problems is a "greater than or equal to" constraint. The first two columns of Table I indicate the coefficient values and the right hand side value of the extra constraint. The column in Table I entitled "Percent of Increase in Objective Function Value Using the Extra Constraint" specifies the percent change in the objective function value when the extra constraint is added to the underlying transshipment problem. The next column in Table I indicates the proportional increase in the objective function value when the slack variable is pivoted into the optimal basis for the singularly constrained transshipment problem (to change a non-integer solution into an integer solution). (Problem 10 in Table I yields an integer solution without requiring the step.)

The columns of Table I titled "Best Results" contain statistics on the most effective solution approach found for the test problems. The first two columns contain the total solution time and the total number of pivots required to solve the constrained problem. The next two columns indicate the time and number of pivots spent searching for the value of the dual variable associated with the extra constraint (3) using the standard Lagrangean relaxation approach [8,9,11,22]. This value was allowed to deviate from a global optimum by at most one unit. That is, upon incorporating the weighted constraint into the objective function, the value of the dual variable is sequentially increased or decreased according to whether the constraint is under or over-satisfied at optimality, until the under and over estimates of the dual variable are within one unit. The standard one-dimensional Golden Search Rule [34] is used to find these estimates. The last spanning tree basis of the search is then augmented to include the slack variable or an artificial variable of the extra constraint (as appropriate), whereupon the solution of the constrained problem

TABLE I
COMPUTATIONAL STATISTICS ON
500 NODE, 2500 ARC PROBLEMS

EXTRA CONSTRAINT (3)							BEST RESULTS				
COEFFICIENT VALUES F_{ij}	RIGHT HAND SIDE VALUE	PERCENT OF INCREASE IN OBJECTIVE FUNCTION VALUE USING THE EXTRA CONSTRAINT	INTEGER OBJ. FUNCTION INCREASE OVER CONTINUOUS OPTIMUM	TOTAL SOL. TIME IN SECONDS*	NO. OF PIVOTS	LAGRANGEAN SEARCH TIME	NO. OF PIVOTS IN LAG. SEARCH	TRANSSHIPMENT SOL. TIME	NO. OF PIVOTS	TOTAL SOL. TIME IN SECONDS*	NO. OF PIVOTS
+1	5,000	4	.00049	6.1	2789	6.1	2789	3.6	2019	35.8	5911
+1	12,500	30	.00083	7.9	3481	7.7	3404	3.6	2019	47.1	733[illegible]
+1	20,000	55	.00188	10.6	3643	10.3	3627	3.6	2019	48.5	8394
1-5 integer	12,500	5	.00078	14.1	5274	12.9	5166	3.6	2019	33.0	5401
1-5 integer	30,000	10	.00145	13.2	4910	12.2	4610	3.6	2019	46.4	7072
1-5 integer	50,000	30	.00098	12.3	4562	11.4	4427	3.6	2019	48.7	7896
1-5 real	6,000	12	.00313	9.8	3742	9.1	3712	3.6	2019	28.4	4482
1-5 real	12,000	30	.00176	9.2	3152	6.9	2889	3.6	2019	26.6	3841
1-5 real	18,000	70	.00296	9.1	3417	8.2	3388	3.6	2019	31.3	4799
-1 or +1	1,250	1	.00000	9.3	3531	8.4	3496	3.6	2019	14.6	3472
-1 or +1	2,500	2	.00116	6.8	2673	6.3	2609	3.6	2019	23.8	4551
-1 or +1	6,000	10	.00219	7.9	3093	7.1	2993	3.6	2019	19.7	4777

*All times are in seconds using a CDC 6600 and a Run FORTRAN compiler is initiated.

The next to last pair of columns of Table I indicate the total solution time and total number of pivots required to solve the underlying transshipment problem. These are based on a one pass "modified row minimum start" [18], and a dynamic candidate list, outward-node most negative pivot rule [15,30]. These procedures have proven to be the most efficient for solving transshipment problems [15,24,25,29,30].

The last two columns of Table I indicate typical results obtained by various alternative solution strategies that we tested for the constrained problem. These results were obtained from a variety of approaches that begin with a modified row minimum start and augment the basis with an appropriate slack or artificial variable at a strategically selected stage of the calculation. As indicated by the results in Table I , the major drawback of this class of strategies is the large number of pivots required to solve the problem. (A large number of these pivots were degenerate.)

Ten different types of start and pivot rules were tested with the alternative strategies. None of these rules substantially reduced the number of pivots. The best of these approaches was to "introduce" the extra constraint to the optimal basis for the underlying transshipment problem. Surprisingly, this approach always dominated an approach which introduced the extra constraint immediately upon encountering a basis in which it became satisfied.

A significant factor in favor of using the Lagrangean approach is that most of the pivots are transshipment type pivots, and hence are much faster to make. In particular, our results indicated that these pivots are about 3 times faster than "case 2b" pivots (See section 4.).

Table II
COMPUTATIONAL RESULTS ON LARGE PROBLEMS

Problem Size Nodes	Arcs	Total Solution Time in Seconds	Number of Pivots	Transshipment Solution Time	Number of Pivots
1000	5000	25.1	5583	12.4	4023
1000	5000	25.6	5732	13.6	4378
1000	5000	27.2	6113	13.5	4561
1000	8000	28.1	6753	14.2	4214
1000	15000	24.2	5341	13.1	3918
2000	6000	65.8	6647	30.1	4418
2000	8000	71.4	7543	38.2	5237
2000	10000	68.3	6961	34.7	4726
2000	12000	75.9	8114	41.2	6221
2000	12000	81.9	9228	43.3	7182
3000	8000	161.0	7549	91.3	5989
3000	8000	180.1	8974	88.2	6741
3000	10000	191.2	9114	97.8	6879
3000	10000	196.6	9321	95.1	6752
3000	12000	277.3	10251	153.4	8188

Computational results demonstrating the effectiveness of I/O PNETS-I on larger problems are shown in Table II. For instance, the solution time for the 3000 node, 12,000 arc singularly constrained transshipment problem is 277 seconds or about 5 minutes. This is an LP problem of considerable size, involving 3001 constraints and 12,001 variables (plus 12,000 upper bound constraints).

In general, our testing indicates that a singularly constrained transshipment problem requires approximately twice as much time to solve as the underlying transshipment problem. Thus, it appears that I/O PNETS-I is at least 75 times faster than a general purpose LP code for solving such problems.

Other findings of our study include the following:

1. The I/O PNETS-I code requires only about 10% more solution time to solve a network problem that has no extra constraint than the I/O PNET-I code which is expressly designed for the pure network problem.

2. The number of non-zero coefficients in the extra constraint affects the number of times the dual variable δ changes and the number of "case 2b" pivots that are performed. Consequently, solution time tends to increase as the number of non-zero coefficients increase.

9.0 <u>SUMMARY</u>

We have shown that the class of singularly constrained network problems, which includes a variety of important practical applications beyond the range of pure network problems, can be solved on a highly efficient basis. The I/O PNETS-I code developed in this study is capable of handling large-scale problems whose dimensions are well beyond the scope of existing LP codes (e.g., involving hundreds of thousands of variables), and has solved a 12,000 variable problem with 3,000 constraints (nodes) in under

5 minutes. Testing also indicates that this code can obtain integer solutions that lie on the average within .007 of the continuous optimum with negligible additional effort.

References

1. Balas, Egon, "Discrete Programming by the Filter Method," Operations Research 19-5, 915-957 (September-October 1967).

2. Barr, R.S., Glover, F., and Klingman, D., "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes." Mathematical Programming,

3. Charnes, A., Lecture notes for Extremal Methods at Carnegie-Mellon, Purdue, Northwestern University and The University of Texas at Austin.

4. Charnes, A. and Cooper, W.W., Management Models and Industrial Applications of Linear Programming, Vols. I and II. New York: John Wiley and Sons, Inc., (1961).

5. Charnes, A., Glover, F., and Klingman, D., "The Lower Bounded and Partial Upper Bounded Distribution Model," Naval Research Logistics Quarterly, Vol. 18, No. 2 (June, 1971), 277-281.

6. Charnes, A. and Klingman, D., "The Distribution Problem With Upper and Lower Bounds on the Node Requirements," Management Science, Vol. 16, No. 9 (May, 1970), 638-642.

7. Dantzig, G.B., Linear Programming Extensions, Princeton, N.J.: Princeton University Press, (1963).

8. Everett, H., "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research 11, 399-417 (1963).

9. Frank, M., and Wolfe, P., "An Algorithm for Quadratic Programming," Naval Res. Log. Quart., 3, 95-110, 1956.

10. Geoffrion, A.M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research 17-3, 437-454 (May-June 1969).

11. Geoffrion, A.M., "Generalized Lagrange Relaxation for Integer Programming," Western Management Science Institute, UCLA, (1974).

12. Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research 13-6, 879-919 (November-December 1965).

13. Glover, F., "Surrogate Constraints," Operations Research 16-4, 741-749 (July-August 1968).

14. Glover, F., "Surrogate Constraint Duality In Mathematical Programming," Operations Research, 73-5, May (1973).

15. Glover, F., Karney, D., and Klingman, D., "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code." Networks, Vol. 20, 191-212, (1974).

16. Glover, F., Karney, D., and Klingman, D., "The Augmented Predecessor Index Method for Locating Stepping-Stone Paths and Assigning Dual Prices in Distribution Problems," Transportation Science, Vol. 6, No. 2 171-179, (May, 1972).

17. Glover, F., Karney, D., Klingman, D., and Napier, A., "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," Management Science, Vol. 20, No. 5, 793-813, (January, 1974).

18. Glover, F., and Klingman, D., "A Note on Computational Simplifications in Solving Generalized Transportation Problems," Transportation Science, Vol 7, No. 4 (November, 1973).

19. Glover, F., Klingman, D., and Ross, G., "Finding Equivalent Transportation Formulations for Constrained Transportation Problems," Naval Research Logistics Quarterly, Vol. 21, No. 2, 247-253, (1974).

20. Glover, F., Klingman, D., and Stutz, J., "Augmented Threaded Index Method For Network Optimization," INFOR, Vol. 12, No. 3, 293-298, (October, 1974).

21. Greenberg, Harvey J., and Pierskalla, W.P., "Surrogate Mathematical Programs," Operations Research, 18, 924-939, (1970).

22. Hogan, W.W., "Convergence Results for Some Extensions of the Frank-Wolfe Method," Working Paper No. 169, Western Management Science Institute, University of California, Los Angeles, January 1971.

23. Johnson, Ellis, "Programming in Networks and Graphs," ORC 65-1, (January, 1965), University of California-Berkeley.

24. Karney, D., and Klingman, D., "Implementation and Computational Study On An In-Core Out-of-Core Primal Network Code." CS 158, Center for Cybernetic Studies, University of Texas, Austin, Texas.

25. Klingman, D., Napier, A., and Stutz, J., "Netgen: A Program for Generating Large Scale Capacitated Assignment, Transportation and Minimum Cost Flow Network Problems," Management Science, Vol. 20, No. 5, 814-821, (January 1974).

26. Klingman, D., and Ross, G.T., "Finding Equivalent Network Formulations For Constrained Network Problems." CS 108, Center for Cybernetic Studies, University of Texas, Austin, Texas.

27. Klingman, D. and Russell, R., "On Solving Constrained Transportation Problems." Operations Research, 23, 1, 91-107, (January, 1975).

28. Klingman, D. and Russell,R., "On Solving Singularly Constrained Transportation Problems." CS 91, Center for Cybernetic Studies, University of Texas, Austin, Texas.

29. McBride, R.D., "Factorization in Large-Scale Linear Programming," Working Paper No. 220, Western Management Science Institute, University of California, Los Angeles, (June, 1973).

30. Mulvey, John, "Column Weighting Factors and Other Enhancements to the Augmented Threaded Index Method for Network Optimization," Joint ORSA/TIMS, San Juan De Puerto Rico, (October, 1974).

31. Shapiro, J.F., "Generalized Lagrange Multipliers in Integer Programming," Operations Research, 19-1, 68-76, January-February 1971.

32. Srinivasan, F. and G.L., Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," JACM, 20, 194-213, (1973).

33. Wagner, Harvey M., Principles of Operations Research, Englewood Cliffs, N.J.: Prentice-Hall, Inc., (1969).

34. Wilde, D.J. and Beightler, C.S., Foundations of Optimization, Englewood Cliffs, N.J.: Prentice-Hall, Inc., (1967).